Scalable Federated Learning Simulations using Virtual Client Engine in Flower

Thomas Borja, Daniel Alamillo, Anass Anhari, Ilker Demirkol
Dept. of Mining, Industrial and ICT Engineering, Universitat Politecnica de Catalunya, Barcelona, SPAIN

***Abstract*—Federated Learning (FL) is a novel technology that has gained significant visibility among the scientific community and industry. In order to be deployed in real scenarios, Federated Learning must address some critical challenges. One key challenge that Federated Learning poses is system scalability, as it is foreseen that this technology will be deployed in IoT scenarios with billions of clients in the future. In addition, there are many ongoing efforts regarding algorithms and frameworks proposed to foster Federated Learning testing and research. In this work, we review a recent architecture of the open-source framework Flower, called Virtual Client Engine (VCE), which allows to efficiently run simulations with hundreds of users with acceptable accuracy and keeping the computational resource consumption low. We compare relevant classification metrics (accuracy, loss) to classic centralized machine learning approaches and Flower's previous architecture named Edge Client Engine (ECE) to show the improvement in the number of clients while keeping the accuracy high. For a 200 clients case, we achieve an accuracy of around 91% for the MNIST and 88% for the SPEECH evaluation setups, both cases using the VCE Flower architecture.**

***Index Terms*—Federated Learning, flower, virtual client engine, classification metrics.**

I. Introduction

While traditional Machine Learning applications use data stored and processed in a centralized fashion, a new paradigm called Federated Learning (FL) aims to work collaboratively with numerous clients who share locally trained models or weights, while keeping their data private [1]. Federated Learning is a promising technology that will allow us to take intelligent services to the next level, as it combines powerful Machine Learning techniques with data generated by users in a ubiquitous way.

Federated Learning comes with critical challenges that must be faced before it can be deployed in real scenarios. One is the massive number of distributed clients participating in the learning process with an order of hundreds in single cluster deployments, thousands in Cross-silo FL, and even millions or billions of connections in Cross-device FL implementations for futuristic networks with massive parallelism, like IoT [2], [3]. Additionally, it is expected that a high number of simultaneous connections would produce bottlenecks or dropouts during the communication process. This problem comes as an inherent issue due to the hardware heterogeneity of devices or slow connections due to bandwidth issues of the users [3].

As an active area of research, there has been a significant contribution to algorithms and frameworks proposed for Federated Learning. In particular, we have chosen Flower [4] for running our metrics evaluation. Flower is an end-to-end open-source Federated Learning framework. Researchers and academics around the world are actively supporting Flower. Furthermore, Flower can run Federated Learning simulations efficiently in a few lines of code for numerous clients. Besides, one crucial feature of Flower is that it can be built on top of other well-known Machine Learning platforms, such as TensorFlow and PyTorch.

To help overcome physical resource consumption and accuracy degradation for simulations involving numerous users, Flower has recently deployed a new system architecture named Virtual Client Engine, which uses a Remote Procedure Call, or RPC, to make virtual clients consume close to zero resources when inactive, and only loading model and data into memory when the client is being selected for training, or evaluation [4]. Virtual Client Engine allows for an exponential increase in the number of participants in the Federated Learning process with efficient hardware consumption while keeping the accuracy high. Additionally, the simplicity of Flower, with the efficient scalability of Virtual Client Engine architecture, allows the deployment of more complex Federated Learning simulations with a high number of users on mid-range computers, and not only in high-capacity supercomputers.

In this work, we propose an evaluation of relevant Federated Learning classification metrics, such as accuracy, loss, and execution time. We focus on a Virtual Client Engine implementation using Flower to show its performance against its previous technology named Edge Client Engine in terms of the number of users, resource consumption, and the trade-off between the number of clients and the accuracy of the trained model for the MNIST dataset. We finally show that using Virtual Client Engine leads to acceptable accuracies of around 91% using 200 simulated clients in Flower.

II. Evaluation of FL metrics with Flower

Flower is a novel open-source framework that allows to deployment of Federated Learning simulations with a few steps and a minimal amount of code to execute. Flower also provides a seamless transition from experimental simulation to actual implementations on real devices, specifically devices with GPU capabilities that have been proven to perform better on Machine Learning technique tasks, e.g., when working with Convolutional Neural Networks for image recognition [4].

Flower allows easy integration with popular machine learning frameworks, such as TensorFlow and PyTorch. Little additional coding knowledge is required to evolve from a classic Machine Learning setup written in Python and TensorFlow to a Federated Learning setup in Flower. In the end, the number of users developing a distributed model depends on the processing capabilities of the computers, servers, or dedicated hardware.

The following subsections describe some critical aspects of the Federated Learning simulations in Flower: System Architecture, the Virtual Client Engine tool, and the Strategies.

1) System Architecture: As in the basic Federated Learning scenario, Flower uses a conceptual and hardware-agnostic architecture consisting of two types of elements or nodes: Aggregator server and collaborator client.

a) Aggregator server: It is the central coordinator in charge of sending a pre-initialized model to the clients at the beginning of the process. Then, the server collects the locally trained models sent by the clients and aggregates them (or averages) using an aggregation algorithm, such as *FedAvg*. In the next cycle step, the server will send the tuned model to each client in an iterative manner until convergence or some fatal error appears.

b) Collaborator client: Each one of the clients only can access their local data. The main task of a client is to locally train the global model received from the Aggregator Server and report back the updates of the training results.

Every local training iteration produced at the client nodes is called an epoch, and every model aggregation produced at the server node is called a round.

2) Virtual Client Engine: Virtual Client Engine is a virtualization tool recently built into Flower that maximizes the available hardware utilization [4]. The VCE handles the scheduling, creation, and running of the Flower Clients in a user-friendly and transparent way for the user and the Flower Server. The VCE efficiently launches Flower Clients based on the pool of clients, their computing and memory needs (such as the number of CPUs and VRAM requirements), and the FL-specific parameters (such as the number of clients per round). This makes parallelizing jobs more manageable and ensures optimal use of the available hardware, allowing the same FL experiment to be used on various setups without making changes - be it a desktop computer, a single GPU rack, or a multi-node GPU cluster. As a result, the VCE becomes a crucial component within the Flower framework, allowing for the efficient execution of large-scale FL tasks with minimal additional effort in a scalable manner.

3) Hardware specifications: A critical feature of the VCE architecture in Flower is that it allows running scalable scenarios with mid-range computers. For the MNIST evaluations, we used a computer with a 2,7 GHz Dual-Core Intel Core i5, 5th generation CPU, with 8 GB DDR3 RAM, running Mac OS. And, for the SPEECH evaluations, we used a computer with 2,4 GHz Intel Core i9, 12th generation, PCU, with GB DDR4 RAM, and Ubuntu 20.04 LTS OS. Although the Flower framework allows CPU and GPU simulations, for this paper we have used only CPU configurations for simplicity.

4) MNIST and SPEECH Federated Setup: For the public MNIST dataset, we ran a Federated Learning scenario with the following conditions: The MNIST complete dataset contains 60.000 training samples and 10.000 testing samples. We use 280 samples per user and a maximum number of 250 clients. From the split dataset, 240 samples are for training, and 40 are for testing per user. Lastly, we use a Convolutional Neural Network machine learning algorithm [5], as it performs well for image recognition.

For the voice command SPEECH dataset, we have 1000 samples for each different command, with a total of 8 commands; this is a total of 8000 samples. We split 30 samples per client with a maximum number of 250 clients. From the 30 samples, 25 are used for training, 2 for validation to prevent overfitting, and 3 for testing purposes. We have used the simple speech recognition model from the Tensorflow official site [6] as the machine learning algorithm.

For both cases, from the entire set of clients selected, the whole group is selected for the FL training, where after each round, half of them will be randomly selected to validate the training. Finally, the number of clients selected for the validation process is half the number of available clients. The complete training process is done in 20 epochs.

5) Flower System Architecture: The following specifications have been used to simulate our scalable Federated Learning scenarios:

a) Using the Edge Client Engine (ECE): The Edge Client Engine from the Flower Architecture will only allow us to run the simulations with a limited number of users, as the hardware resources will be depleted. In addition, in this case, the simulation on the clients should be started manually. A number of 10 clients is selected for this scenario.

b) Using the Virtual Client Engine (VCE): We have seen that using the ECE architecture locally on a single machine having 10 clients means that we are getting 10 instances of the MnistClients parameter, meaning that they will share resources such as CPU, GPU, and RAM. Hence increasing the number of clients instances can quickly exhaust the available resources. However, with the built-in Virtual Client Engine, we can adapt the simulations to run for a more significant number of scenarios. We have selected 50, 100, 150, and 200 simultaneous clients in both cases.

III. Results and Discussion

After simulating various scenarios using the Flower framework, we can discuss some key points: First, we have deployed an Edge Client Engine-based simulation with 10 users and some Virtual Client Engine-based simulations with 50, 100, 150, and 200 clients. While both configurations offer acceptable accuracies, loss curves, and execution times, we have to emphasize that the VCE simulation allows to run considerably more users, which translates to a more realistic scenario. We will direct the discussion to three features for both of the datasets: Classification metrics, Execution Time, and Complexity and Resource Consumption.

We present the results obtained for each one of the scenarios deployed and for the two public datasets used. Each following subsection contains the results obtained for an specific evaluation meter, as well as the discussion for each item.

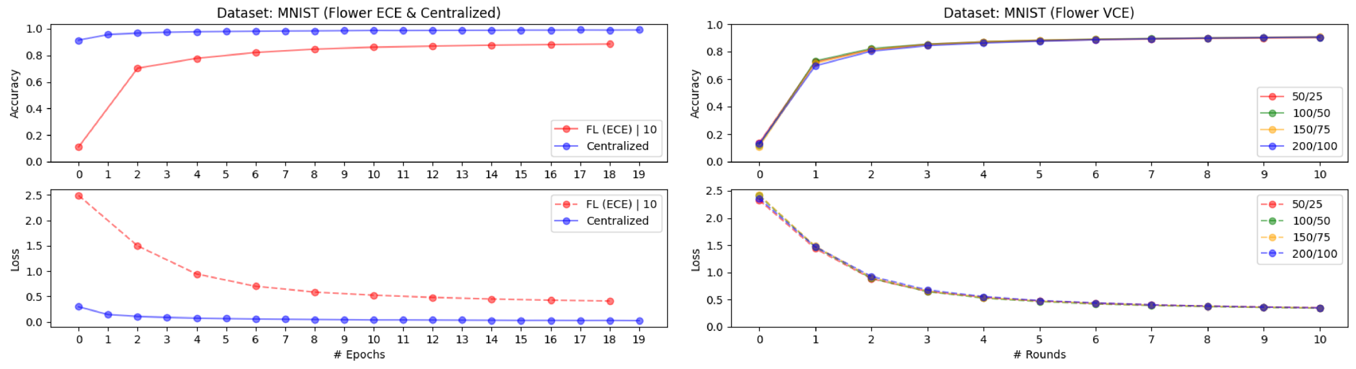

Fig. 1. Classification metrics for the MNIST dataset: Flower ECE, 10 clients (left), and Flower VCE, variable number of clients (right).

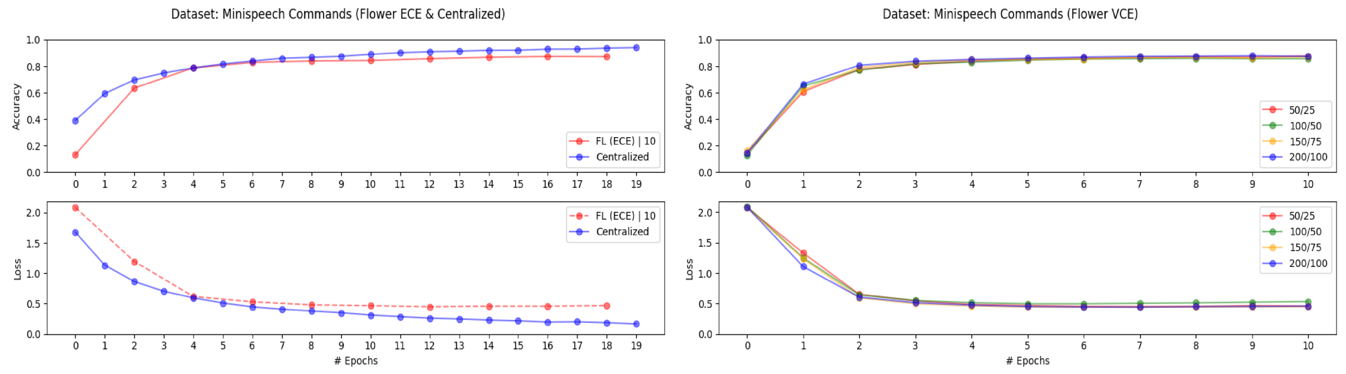

Fig. 2. Classification metrics for the SPEECH dataset: Flower ECE, 10 clients (left), and Flower VCE, variable number of clients (right).

A. Classification metrics

The results shown in Fig. 1 and Fig. 2 summarize the accuracy and loss metrics for the two Federated Learning scenarios. We present the performance results of the three cases implemented: centralized classic ML, Flower ECE with 10 clients, and Flower VCE with a scalable number of users, up to a maximum of 200 clients.

For the MNIST dataset, ECE with 10 clients and VCE with 200 clients have shown acceptable accuracies of 20 epochs for both cases. ECE and VCE reach an accuracy of approximately 89% and 91%, respectively, showing that even with more clients, it is possible to achieve better results. The 150-client scenario shows the best accuracy performance with a difference of some percentage decimals. For the SPEECH recognition dataset, ECE with 10 clients and VCE with 200 clients reach an accuracy of approximately 87% and 88%, respectively. In the second case, the accuracy is lower, and it can vary depending on the complexity of the dataset, the hyper-parameter tuning, or even the type of algorithm chosen for the training.

B. Execution times

For the MNIST dataset with VCE architecture, we present the cumulative training time for 50, 100, 150, and 200 federated clients in Fig. 3. It is relevant to notice that the time employed for training 200 clients is not necessarily the double of the time needed for the 100 client equivalent. Additionally, the time needed to complete the 200 client training is slightly above the previous 150 client cases.

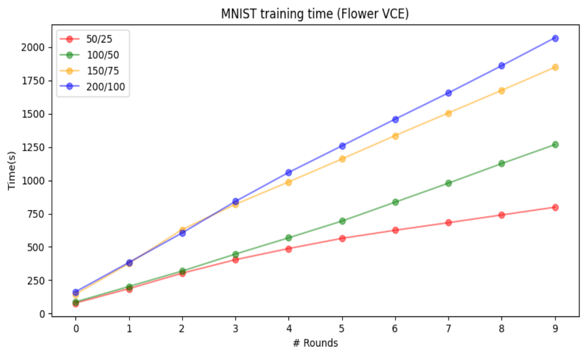

Fig. 3. Federated Learning cumulative training time for MNIST dataset with variable number of users.

We present in Fig. 4 the curve for the cumulative MNIST dataset with the maximum number of clients of 250 and compare it to the ECE case with 10 clients and to the centralized Machine Learning training. Although VCE holds the most simultaneous clients with reduced resource consumption and high accuracy, the total training time for the ECE architecture and classic ML is considerably lower.

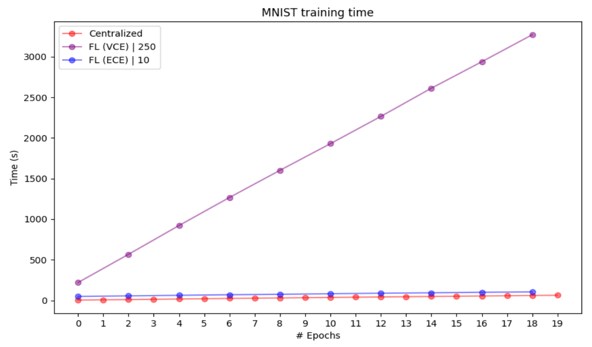

Fig. 4. Federated Learning cumulative training time for MNIST dataset with VCE 250 clients, ECE with 10 clients, and using centralized ML.

For the SPEECH dataset with VCE architecture, we present the total training time for 50, 100, 150, and 200 federated clients in Fig. 5. In addition, Fig. 5 shows the total training time required to complete a classic centralized Machine Learning training and the total time for the ECE scenario with 10 clients, both of the last cases being lower than the VCE architecture. In the classic ML training approach, all the client data is accessible at a centralized machine, where the machine learning model is trained.

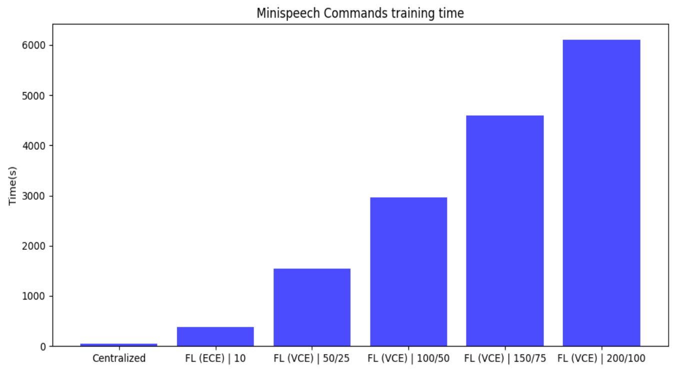

Fig. 5. Federated Learning total training time for SPEECH dataset with variable number of users, ECE case with 10 clients and centralized ML approach.

It is noticeable that the VCE also shows the highest execution times, with around 50 minutes (3000 seconds) for the MNIST dataset and around 100 minutes (6000 seconds) for the SPEECH dataset, both cases with the most complex scenario of 250 clients. Although this time may be considered high, it still benefits from the client scalability feature and keeps the implementation easy. On the other hand, we cannot compare this time to the ECE scenario, as it is only possible to run such several simultaneous clients by exhausting the same physical resources. Even though the execution time of VCE is the best strength of the Flower framework, it remains low compared to other simulations with a high number of users, as stated in the Literature Review subsection.

C. *Complexity and Resource Consumption*

ECE and VCE architectures require little coding and configurations before running simulations in Flower. The ECE architecture needs to be handled with care, as it may cause some crashes when the dataset is not adequately split or when the simulation parameters, such as the number of clients, is too high. Our experiments found that running ECE with 50 clients may cause the RAM to be depleted and freeze the server. Another difference is that for the ECE case, each one of the clients should be started manually, while VCE handles the automation process for the clients. In addition, as ECE showed a higher resource consumption than VCE in processing and RAM, the VCE architecture is better for efficient and quick scalability and lower resource consumption.

Finally, as a benchmark, we ran the simulations for a centralized classic ML learning scenario using the same datasets and similar configurations. Figs. 1 and 2 show that it is possible to achieve a higher accuracy of around 99% in this scenario. This difference of accuracy shows the challenge of FL for the accuracy metric.

IV. Conclusions

The most recent Flower's system architecture, Virtual Client Engine, stands as an attractive option to deploy scalable Federated Learning simulations, achieving decent accuracies with minimum effort and keeping the physical resource consumption low, allowing it to run on regular mid-term computers. Hence, it is reachable to more interested people in the academy and industry. Virtual Client Engine architecture of Flower shows promising results when simulating Federated Learning training with hundreds of clients and with low execution times compared to other studies. Additionally, VCE is agnostic to the dataset used, the Machine Learning algorithm, or the hyperparameter tuning, thus giving it a broad scope of application.

Acknowledgment

This work has been funded partially by the FLAIR and VEDLIoT projects funded from the European Union's Horizon 2020 research and innovation programme under grant agreement No 957197.

References

[1] H. B. McMahan *et al.*, "Communication-efficient learning of deep networks from decentralized data," *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics, AISTATS 2017*, 2017.
[2] W. Huang *et al.*, "Fairness and accuracy in federated learning," 12 2020. [Online]. Available: http://arxiv.org/abs/2012.10069
[3] W. Y. B. Lim *et al.*, "Federated learning in mobile edge networks: A comprehensive survey," *IEEE Communications Surveys and Tutorials*, vol. 22, pp. 2031–2063, 7 2020.
[4] D. J. Beutel *et al.*, "Flower: A friendly federated learning research framework," 2020. [Online]. Available: http://arxiv.org/abs/2007.14390
[5] S. Tabik *et al.*, "A snapshot of image pre-processing for convolutional neural networks: case study of mnist," 2017.
[6] TensorFlow, "Simple audio recognition: Recognizing keywords. url: https://www.tensorflow.org/tutorials/audio/simple_audio," 2023. [Online]. Available: https://www.tensorflow.org/tutorials/audio/simple_audio